REMARKS/ARGUMENTS

The Examiner has indicated in the Office Action summary on page 1 of the Office Action that claims 1-34 are allowed. The indication on this summary page is considered an error in view of the Examiner's comments.

Claims 1-12, 14-21, and 23-34 are pending in the present application. Claims 23 and 28 have been amended. Support for the amendments can be found in claim 22 as originally filed, and later canceled in the response to Office Action filed on September 3, 2003, and in the specification at least on page 71, lines 3-11. Reconsideration of the claims is respectfully requested.

I. 35 U.S.C. § 101

The Examiner has rejected claims 28 and 33-34 under 35 U.S.C. § 101 as being directed towards non-statutory subject matter. In response to the Examiner's comments, claim 28 has been amended to recite that the computer program product is embodied in a computer readable medium. Accordingly, the rejection of claim 28 under 35 U.S.C. § 101 has been overcome. Because claims 33-34 depend from claim 28, the rejection under 35 U.S.C. § 101 as to claims 33-34 has also been overcome by the same amendment to claim 28.

II. 35 U.S.C. § 102, Anticipation

The Examiner has rejected claims 1-10 under 35 U.S.C. § 102(e) as being anticipated by *Guinther* et al., Accurate Profile and Timing Information for Multitasking Systems, U.S. Patent No. 6,016,466 (hereinafter, "*Guinther*"). This rejection is respectfully traversed.

The Examiner has rejected these claims stating:

> As per claim 1, Guinther disclose the "receiving an event indication" (Guinther, col. 21:124-26);
> Guinther also teaches "ascertaining kernel thread level profile information" (Guinther, col. 21:29-33);
> Guinther also discloses "calculating thread level profile information (Guinther, col. 21:1-3);
> Guinther also teaches "identifying a kernel thread" (Guinther, col. 20: 43-48);
> Guinther also discloses "determining whether the identified kernel thread has been reused" (Guinther, col. 21:13-14);
> Guinther also teaches "updating" profile information (Guinther, col. 2:1-3).

Office Action dated October 20, 2006, p. 3.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). In this case, each and every feature of the presently claimed invention is not identically shown in the cited reference, arranged as they are in the claims.

Guinther does not anticipate claims 1-10 as asserted by the Examiner. In particular, claim 1 recites:

> A method for monitoring performance of a program being executed using per thread metric variables with reused kernel threads comprising:
> receiving an event indication;
> ascertaining kernel thread level profile information;
> identifying a kernel thread, wherein the kernel thread level profile information is attributed to the identified kernel thread;
> determining whether the identified kernel thread has been reused; and
> updating profile information with the kernel thread level profile information based whether the identified kernel thread has been reused.

Guinther does not teach or suggest the receiving, identifying, determining, and updating steps as recited in claim 1. As to the receiving step in claim 1, the Examiner cites the following section from *Guinther* as teaching this step:

> Processing begins a first test step **452** where it is determined if the thread being swapped in is being monitored by the profiler **402.**

Guinther, col. 21, ll. 24-26.

The cited section describes determining if a thread that is being swapped is being monitored by a profiler, which is not the same as receiving an event notification. No teaching is present for the step of receiving an event indication in the section cited by the Examiner. However, this section should not be analyzed in isolation because other sections of *Guinther* provide the context necessary for proper interpretation of the cited section. In this section, *Guinther* describes a determining step in *Guinther's* process, depicting the operation of an operating system's swap context routine that has been patched to make a call into a device driver. This analysis is supported by *Guinther* as follows:

> Note that the parameters to the swap context routine include an identifier of the thread being swapped in as well as an identifier of the thread being swapped out. Thus, at the test step **452,** the identifier of the thread being swapped in is compared with the thread identifiers stored in the thread database **412** and described above in connection with FIG. **13.** If the thread being swapped in is found in the thread database **412,** then control passes from the test step **452** to a step **454** where the device driver **410** obtains the system time.

Guinther, col. 21, ll. 27-36.

In this additional section, the workings of the determining step are described in a more complete fashion. Here, *Guinther* describes that the determining step is really a comparison step where the identifier of the thread being swapped in is compared to the entries of thread identifiers in a database. Thus, the Examiner's assertion that *Guinther* teaches the receiving step as recited in claim 1 equates the comparison of an identifier with entries in a database, with receiving an event indication. Such an interpretation is not taught by this cited reference and the Examiner has not pointed out any teaching elsewhere in the prior art for this interpretation. *Guinther* teaches the former, claim 1 recites the latter, and consequently, *Guinther* fails to teach the claimed feature.

Guinther's disclosure, as applied by the Examiner, fails to teach the receiving step for at least two reasons. First, comparison of one item of data with another is a process, and not an event indication as recited in claim 1. Presumably, comparison of an item of data with a collection of similar data items is a repetitive database process to some extent, according to *Guinther's* disclosure. For this reason, neither the cited section, nor the remainder of *Guinther's* disclosure teaches a discrete event indication as recited in claim 1. For a reference to anticipate a claimed invention, the reference must teach each and every element of the claim, arranged identically as claimed. *In re Bond. Guinther* does not teach an event indication as recited in claim 1 in an arrangement identical to the arrangement of features in this claim.

Second, *Guinther's* disclosure does not teach "receiving" of anything, much less receiving an event indication as recited in claim 1. Again, the section cited by the Examiner should not be analyzed in isolation for this second reason. The following additional section and figures provide the necessary context for proper analysis of *Guinther*'s teachings:

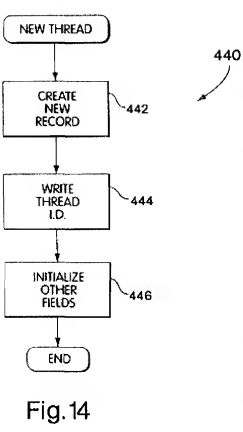

Fig. 14

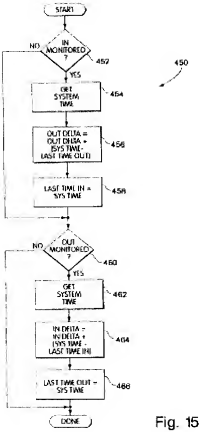

Fig. 15

Guinther, Figures 14-15.

These figures are analyzed here only to the extent they are needed to supply the necessary context for analyzing the section cited by the Examiner. Figure 14 illustrates a process in *Guinther* by which a record of a thread is created. Among other things, an identifier of the thread is written into the record when a new record for a new thread is created.

Figure 15 then begins with the inspection of the record created and populated in Figure 14. Step 452 compares the thread identifier that has been written into this record in Figure 14, with a database of thread identifiers. The following section supports this analysis of the two figures:

> Processing begins at a first step **442** where a new record is created for the thread. The record is similar to those records **432-434** shown in FIG. **13.** Following step **442** is a step **444** where the thread i.d. is written into the record. In some instances, the thread i.d. provided by the operating system may be mapped into a more convenient identifier. Following the step **444** is a step **446** where the other values of the record are initialized, for example, to zero.
>
> Referring to FIG. **15,** a flow chart **450** shows operations that are performed in connection with an operating systems swap context routine that has been patched to make a call into the device driver **410,** as discussed above. Processing begins a first test step **452** where it is determined if the thread being swapped in is being monitored by the profiler **402.**

Guinther, col. 21, ll. 13-26.

Again, for the limited purpose of providing sufficient context to the section cited by the Examiner, this section of *Guinther* provides that a new record is created for a new thread. The record is then populated with thread identifiers and other values pertaining to the new thread in subsequent steps of the process of Figure 14. The above section then continues to describe how the thread identifier in this record is compared with thread identifier entries in a database.

With this contextual information, note that the record has not been sent or received at all. The record is simply referenced multiple times for various process steps. Consequently, there is no "receiving" of the record, or of the thread identifier therein, in any manner than can be considered equivalent to "receiving an event indication" as recited in claim 1. Thus, for these two reasons, contrary to the Examiner's assertion, *Guinther* fails to teach the receiving step as recited in claim 1.

The Examiner further rejects claim 1 by stating that *Guinther* teaches a "calculating" step that is allegedly recited in claim 1, whereas claim 1 recites no such step. Applicants believe that the Examiner is referring to the claim 1 step that recites "identifying a kernel thread, wherein the kernel thread level profile information is attributed to the identified kernel thread" when rejecting the "calculating" step, and address the rejection accordingly.

With respect to the identifying step of claim 1, the Examiner cites to a section of *Guinther* that is made up of two partial sentences from two different paragraphs that are incoherent as cited. A larger section surrounding the Examiner's cited section is quoted and analyzed below. The Examiner incorrectly interprets that *Guinther* teaches the step in the following section:

> The thread database **412** is thus a collection of records **432-434** for each of the threads that is being profiled. Note that the records **432-434** may contain other data fields and that the records **432-434** may be stored in the thread database **412** as an array, a linked list, or by using any other appropriate, conventional data storage mechanism.
>
> Referring to FIG. **14,** a flow chart **440** indicates operations performed by the device driver **410** when it receives an indication that a new thread is being created via the DeviceIOControl call.

Guinther, col. 20, l. 64 – col. 21, l. 6.

As can be seen, this section of *Guinther* describes the database discussed above as a collection of thread records. The section further describes that the records may be organized in a variety of data structures to constitute that database. Finally, the section describes when the process of Figure 14, analyzed above, is started.

Absent in this section is the disclosure of an "identifying" step as claimed in claim 1. The section cited by the Examiner, together with surrounding information from the larger section described above, is

largely irrelevant to the claim feature at hand. Organization is not the same as identifying, and organizing records in a data structure does not teach "identifying a kernel thread, wherein the kernel thread level profile information is attributed to the identified kernel thread" as claimed. Therefore, *Guinther* fails to teach the "identifying" step as recited in claim 1.

As to the determining step of claim 1, the Examiner incorrectly asserts that *Guinther* teaches this step in the following section:

> Processing begins a first test step **452** where it is determined if the thread being swapped in is being monitored by the profiler **402.**

Guinther, col. 21, ll. 13-14.

This cited section describes the beginning of the process of Figure 14, which has already been described in detail above. The teachings of this section are limited to informing that a new record is created for a new thread. As described above with respect to other parts of *Guinther's* disclosure, this new record for the new thread is then populated with thread identifiers and other thread related values. Note, a new thread is created and a new record is created for the new thread, as clearly depicted in *Guinther*'s Figure 14, reproduced above. A new thread is not a reused thread. The section cited by the Examiner, other relevant contextual information pertaining to the cited section, and Figure 14 to which the cited section relates, are all devoid of any teaching of "determining whether the identified kernel thread has been reused", in the manner recited in claim 1. Therefore, contrary to the Examiner's assertion, *Guinther* fails to teach the determining step of claim 1 as well.

As to the updating step, the Examiner cites the following section as teaching this step:

> The expected execution time may be determined using the individual execution times of each of the instructions that make up the block of code.

Guinther, col. 2, ll. 1-3.

This section is irrelevant to claim 1 in general, and the updating step of claim 1 in particular. It is common knowledge that the execution time for a block of code is the total time of execution of the individual instructions comprising the block of code. The section cited by the Examiner appears to be a random citation, which states a truism, and is irrelevant to the claim feature at hand. Therefore, the Examiner has failed to cite a proper teaching in the reference as to the updating step of claim 1.

For the foregoing reasons, *Guinther* fails to anticipate claim 1 under 35 U.S.C. § 102(e) as asserted by the Examiner. Consequently, the rejection of claim 1 has been overcome. Dependent claims 2-10 are also not anticipated by *Guinther*, at least by virtue of their dependence from claim 1.

The dependent claims are further not anticipated by *Guinther* because they contain additional features that are not taught by *Guinther*. For example, claim 3 recites:

> The method recited in claim 1 above, wherein updating profile information with the kernel thread level profile information based on the identified kernel thread is reused further comprises:
> applying the kernel thread level profile information for the reused identified kernel thread to one of a previous application thread and a new application thread.

The Examiner states:

> As per claim 3, as applied to claim 1 above, Guinther teaches "applying the difference kernel thread level profile information . . . to one of a . . . thread" (Guinther, col. 21, line 64, to col. 22, line 11).

Office Action dated October 20, 2006, p. 4.

Contrary to the Examiner's assertion, *Guinther* does not teach the applying step of claim 3. Particularly, within the applying step, *Guinther* fails to teach "applying the kernel thread level profile information for the reused identified kernel thread to one of a previous application thread and a new application thread". The Examiner cites to the following section of *Guinther* as teaching the step:

> If it is determined at the test step **460** that the thread being swapped out is being monitored, then control passes from the test step **460** to a step **462** where the device driver **410** obtains the system time in a manner similar to obtaining the system time at the step **454,** discussed above. Following step **462** is a step **464** where the indelta value is incremented by an amount equal to the system time minus the last time in.
> Note that since the task is being swapped out, then the quantity system time minus lasttimein reflects the amount of time that the thread has spent being swapped in. Thus, the new value of indelta is incremented by this amount of time that the thread has spent being swapped in for this iteration. Following the step **464** is a step **466** where the lasttimeout is set equal to the current system time. Following the step **466,** processing is complete.

Guinther, col. 21, l. 64 – col. 22, l. 11.

This section essentially teaches how to compute the time a thread spends executing. This section describes a swapped in thread, which is the new thread of *Guinther's* Figure 14, and whose identifier is determined to be matching with an identifier stored in a thread database in *Guinther's* Figure 15.

Even if, *arguendo*, the thread execution time is considered an information equivalent to the information as claimed, this section fails to teach that the execution time is the information of a reused thread. The section further fails to teach that the information is applied to a previous, or a new application thread. In order to anticipate a claim, a reference must teach each and every element of the

claim arranged identically as in the claim. *In re Bond*. *Guinther* fails to teach each and every element of claim 3, particularly failing to teach the nature of the thread to which the information pertains, as recited in this claim. Therefore, for this additional reason, *Guinther* fails to anticipate dependent claim 3.

As a second example of a dependent claim that is not anticipated by *Guinther*, contrary to the Examiner's assertions, *Guinther* fails to teach the associating step as recited in claim 4.

The Examiner states:

> As per claim 4, as applied to claim 2 above, Guinther discloses "applying the difference kernel thread level profile information . . . to one of a . . . thread" (Guinther, co. 22, 1-8);
>
> Guinther also teaches "marking the previous Java thread being terminated" (Guinther, col. 22: 9- 11);
>
> Guinther also discloses "associating the reused identified kernel thread with the new Java thread in the hash table" (Guinther, col. 21: 15 -1 8).

Office Action dated October 20, 2006, p. 4.

Claim 4 recites:

> The method recited in claim 2 above, wherein updating profile information with the kernel thread level profile information based on the identified kernel thread is reused further comprises:
>
> applying the kernel thread level profile information for the reused identified kernel thread to one of a previous application thread and a new application thread;
>
> marking the previous application thread being terminated; and
>
> associating the reused identified kernel thread with the new application thread in the hash table.

The Examiner cites to the following section of *Guinther* as teaching this step:

> Following step **442** is a step **444** where the thread i.d. is written into the record. In some instances, the thread i.d. provided by the system may be mapped into a more convenient identifier.

Guinther, col. 21, ll. 15-18.

This section describes yet another step, step 444, in *Guinther's* Figure 14. In this step, *Guinther* teaches that the thread identifier is written into the record. As already described in the foregoing analysis, the thread to which this step in Figure 14 pertains is a new thread. Furthermore, the only teaching present in the cited section is that the new thread's identifier is written into the record, not associating the reused identified kernel thread with the new application thread as recited in claim 4. *Guinther* neither teaches two threads – a reused thread and a new thread, nor teaches reusing and rewriting a record because Figure 14 clearly indicates that the record being written is a new record. Therefore, contrary to the Examiner's assertion, *Guinther* fails to teach at least the associating step of claim 4. Thus, the rejection of claims 1-

10 under 35 U.S.C. § 102(e) as being anticipated by *Guinther* has been overcome.

III. 35 U.S.C. § 103, Obviousness

The Examiner has rejected claims 12-32 under 35 U.S.C. § 103(a) as being unpatentable over *Guinther*, in view of Java Virtual Machine Profiler Interface (JVMPI) http://java.sun.com/pioducts/jdk/1.2/docs/guide/jvmpi/jvmpi.html, A Sun Microsystems Corporation's web publication (hereinafter "*Java*"). This rejection is respectfully traversed.

The Examiner states:

> As per claim 12,18, 19, and 26, Guinther teaches, by implication, the "receiving a value of a metric variable for a kernel thread" and the "calculating a difference value for the metric variable" (Guinther, col. 22:1-8);
>
> Guinther also discloses "determining if the kernel thread has been used" (Guinther, col. 21:59-61);
>
> Guinther does not expressly disclose "applying the difference value of the metric variable to a Java thread."
>
> However, the JVMPI does disclose the application of a metric variable difference to a Java thread record (JVMPI, pg. 35). Thus, at the time the invention was made, it would have been obvious to a person of ordinary skill in the art that the profiling of Guinther could be combined with the Java thread profiling of JVMPI. One of ordinary skill in the art would have been motivated to do this in order to obtain run time information for a Java program.

Office Action dated October 20, 2006, p. 6.

The Cited References Do Not Teach All of the Features of Claim 12

The Examiner has failed to state a *prima facie* obviousness rejection because the cited references used in the proposed combination do not teach all of the features of claim 12 as believed by the Examiner. Applicants distinguish the references from claims 12-32 using claim 12 as an example. The arguments below as applied to claim 12 are similarly applicable to claims 14-32. Claim 12 recites:

> A method for monitoring performance of a program being executed using per thread metric variables with reused kernel threads comprising:
>
> receiving a value of a metric variable for a kernel thread;
>
> determining if the kernel thread has been previously used by a first application thread; and
>
> applying the value of the metric variable to a second application thread if the kernel thread has been previously used by the first application thread.

A *prima facie* case of obviousness is established when the teachings of the prior art itself suggest the claimed subject matter to a person of ordinary skill in the art. *In re Bell*, 991 F.2d 781, 783, 26 U.S.P.Q.2d 1529, 1531 (Fed. Cir. 1993). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). In the case at hand, not all of the features of the claimed invention have been considered and the

teachings of the references themselves do not suggest the claimed subject matter to a person of ordinary skill in the art.

Contrary to the Examiner's assertion, *Guinther* does not teach or suggest the determining step as recited in claim 12. The Examiner cites the following section of *Guinther* as teaching or suggesting this step:

> Following the step **458** is a step **460** where it is determined if the thread being swapped out is a thread that is being monitored.

Guinther, col. 21, ll. 59-61.

This section of *Guinther* describes a step in the process of Figure 15, which has been reproduced above. Here, *Guinther* informs that the process checks to see if the thread being swapped out is a thread that is being monitored. This section and surrounding relevant information in *Guinther*'s disclosure teach or suggest nothing that corresponds to determining if a kernel thread has been previously used by a first application thread.

The teaching or suggestion asserted by the Examiner utilizing this section pertains to determining whether the thread is monitored, not whether the thread has been previously used. The two determinations are vastly different from one another. Determining whether something is monitored does not teach or suggest determining whether that thing has been previously used. By this reasoning, *Guinther* fails to teach or suggest the determining feature as recited in claim 12.

Therefore, the Examiner is incorrect in alleging that *Guinther* teaches all but the applying step of claim 12. The Examiner has conceded that *Guinther* does not teach or suggest the applying step. As described above, *Guinther* also does not teach the determining step. Consequently, whether *Java* teaches or suggests the applying step of claim 12 is moot in view of this deficiency. Because the Examiner has cited *Java* only to find support for the applying step of claim 12, the combination of *Guinther* and *Java* is insufficient for teaching or suggesting all the features of claim 12.

Independent claims 19, 21, and 28 contain features similar to those in claim 12, and are not made obvious by similar reasoning. Dependent claims 14-18, 20, 23-27 and 29-32 are also not made obvious by *Guinther* alone, or *Guinther* in view of *Java*, at least by virtue of their dependence from one of these independent claims. Therefore, the rejection of claims 12, 14-21 and 23-32 under 35 U.S.C. § 103(a) has been overcome.

IV. Conclusion

It is respectfully urged that the subject application is patentable over *Guinther* and *Java* and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: January 19, 2007

Respectfully submitted,

/Rakesh Garg/

Reg. No. 57,434
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Agent for Applicants